1996

VERIFICATION OF A PC/WINDOWS VERSION OF THE TACOM/TARDEC ACOUSTIC DETECTION RANGE PREDICTION MODEL (ADRPM) TRANSLATED FROM THE HP-UX VERSION

Mr. Roger Evans
Mr. Robert M. Mantey, Jr.
Mr. David Thomas

U.S. Army TARDEC
Warren, Michigan, 48092

ABSTRACT

ADRPM (Acoustic Detection Range Prediction Model) is a software program that models the propagation of acoustic energy through the atmosphere and the detectability of that energy. This model was recently rewritten from a HP-Unix version into a PC/Windows version. This paper describes the software testing methodology used to verify that the new version provides the same results as the old version.

History of ADRPM

ADRPM began its existence in the 1970s as a simple BASIC program. Table I shows its evolution from ADRPM I to ADRPM VII, and its evolution through various platforms and programming languages. Much of the past development on ADRPM has been performed by the BBN Corporation. The version in recent use runs only on 68040-based HP-9000 Unix workstations.

Version	Year	Comments
I	1974	BASIC, by BBN Corp.
II	1975	BASIC, BBN
III	1977	BASIC, BBN
IV	1978	BASIC, paper tape support added by TACOM
V	1980	FORTRAN, University of Dayton
VI	1983	FORTRAN, Wyle Research, BBN, Prime 850 computer
VII	1988	C, BBN, PC/DOS version
VII	1993	C, BBN, 68040 HP-9000 Unix version
VII	2000	C, TACOM, PC/Windows version

Table I – Evolution of ADRPM

Capabilities of ADRPM

ADRPM has 4 primary calculations, given a certain vehicle signature, environmental conditions, and detector parameters:

1) Calculate the acoustic detection range of the vehicle
2) Given a detection distance, calculate the acoustic signature level which may not be exceeded
3) Given a distance, show the vehicle's acoustic signature, independent of detectability
4) Provide a sensitivity analysis capability, where the effect of sweeping one of the input parameters over a range can be analyzed.

Each calculation above then has 3 graphical outputs, a source spectrum, a signal-to-noise spectrum, and an attenuation spectrum. The results specific to each calculation are also presented separately in dialog boxes, such as the detection range or allowable signature level in dB.

A number of inputs are required in order to accomplish the above tasks, for the source these include:

- one-third-octave band signature values (10 Hz to 10 kHz, 31 bands)
- reference distance (source-to-microphone)
- reference height
- reference temperature, humidity, and flow resistance.

Inputs for the propagation path are:

- one-third-octave ambient noise spectrum
- temperature, humidity, wind, weather condition
- flow resistivity
- surface roughness
- barriers (distances and heights)
- foliage bands (distances and widths)

For the detector parameters:

- one-third-octave noise floor
- detector height
- probability of hit
- probability of false alarm
- detector efficiency
- detector type (human or ideal)
- detection rule (DPMAX or DPSS, described below)

Only a single, omnidirectional transducer or single human observer is modeled, and for the human detection case, the lower frequency bound is 40 Hz. The DPMAX, or d'_{max} detection rule means a single band over a certain threshold is the detection case. The DPSS, or d'_{sum} rule uses a sum of squares of detectabilities in all one-third-octave bands as its detection case. DPMAX is recommended for naïve human observers, and DPSS for skilled human observers or electronic detection systems.

Actually, multiple detection ranges are possible, such as the case where a barrier makes a vehicle undetectable for a certain range, but further out the vehicle is again detectable. ADRPM searches for solutions out to 20 km.

The exact algorithms used in ADRPM have varied over the years, and vary over the spectral bands. Details on the techniques can be found in (ref. 1). In any case, the following factors are considered in the propagation model:

- geometric spreading ($^1/_r{}^2$)
- atmospheric absorption ("classical" and molecular)
- refraction (due to temperature and wind)
- ground impedance
- surface roughness
- barriers and foliage bands.

<u>A Windows Version of ADRPM</u>

While ADRPM has been in continual use to the present day, it has recently only been running on the Hewlett-Packard HP-9000 Unix workstation. With the encroaching obsolescence and rarity of this hardware platform, we decided to port the program over to the PC/Windows platform. Borland C++ Builder 4.0 was used for development due to its allowing relatively easy creation of GUI (graphical user interface) programs, and because the existing ADRPM was written in C. The new version runs under Windows 95, 98, NT, and 2000.

Figure 1 shows the main screen of the HP version and Figure 2 that of the Windows version for comparison.

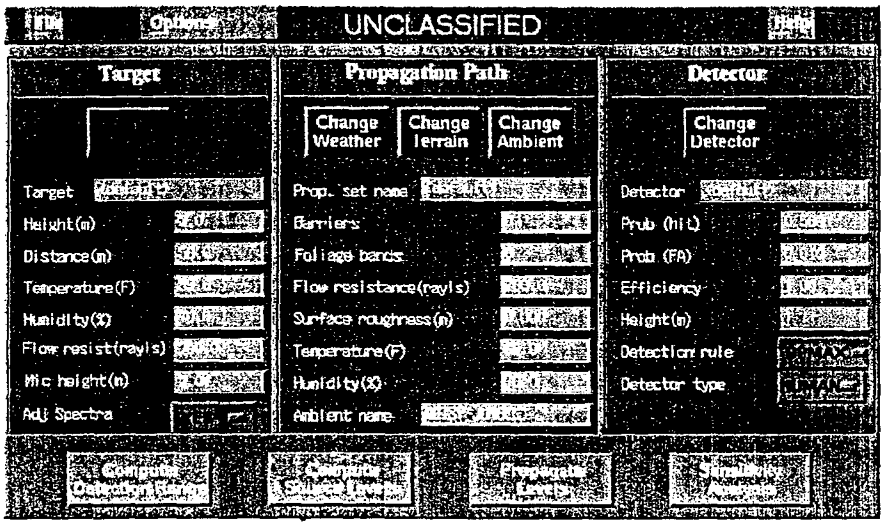


Figure 1 – HP/Unix ADRPM Main Screen

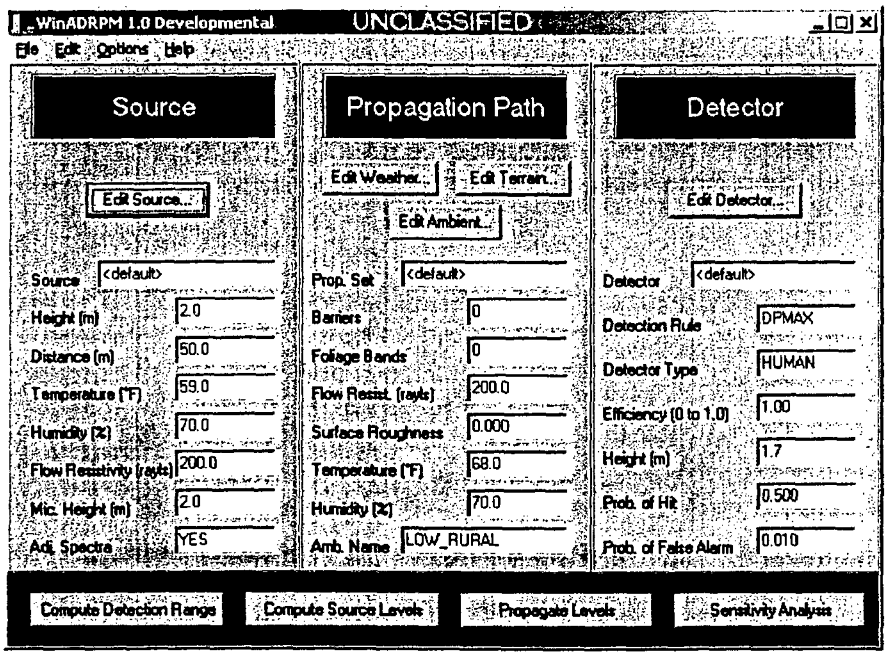


Figure 2 – PC/Windows ADRPM Main Screen

Once this new software was completed, the obvious question was, "Can we get rid of the old HP hardware now?" Despite many successful runs with the default input data values, minor variations from these default values, and some testing with field test data, we could never quite bring ourselves to shut down the HP system for the last time and send it on its way.

Even though the core calculation code of ADRPM is separate from the GUI (graphical user interface), there is still a certain amount of data validation, preprocessing, and postprocessing going on in the GUI code, and all this had to be rewritten for the Windows version. Although ADRPM isn't considered to be a "validated" model, it has been run by various users for many years with good results, and most of its algorithms have been individually validated with real-world testing. We needed to make sure our new Windows version retained the user confidence level provided by the well-tested HP-Unix version.

Software Verification Method

We needed a collection of input files (source, propagation, and detector) which would exercise all of the input variables over their full ranges, and in various combinations. This method is referred to in software engineering as statistical testing. Unfortunately, many of ADRPM's inputs are floating-point numbers with a wide range. Even if, for example, 20 values are used for each, a full combinatorial test using every possible combination of inputs quickly requires millions of runs. We settled upon a testing method that exercises all of the input variables over their full ranges, tests many combinations of these variables, but which does not require an impossible number of runs.

A simple command-line program written using about 750 lines of C++ code reads in script files and generates ADRPM input files with randomized values. Three types of input files are required for a unique ADRPM run: source, propagation (weather, terrain, and ambient conditions), and detector. The generator program reads in these script files and is capable of generating all 3 types of ADRPM input file.

```
# srcrun1.txt
# NOTE: order of items must match order of .libsrc file
#
# will generate srcrun1_1.libsrc, srcrun1_2.libsrc...
#
filename n srcrun1_ libsrc
trgnam l test_source
refdis f 0.0 50000.0 5
reftmp f -50 120 5
refhum f 0 100 5
trghgt f 0 100 5
refflwres f 0 40000 5
michgt f 0 20 5
adjust b NO YES
sspec  c srclist1.txt 5
```

Figure 3 – Example Script File for Generating Source Input Files

Figure 3 shows an example script file which generates source (.libsrc) input files. The line "refhum f 0 100 5" indicates we want to generate at least five floating-point values for the refhum (reference humidity) global variable, each between 0 and 100. As currently implemented, the program will use 5 values evenly spaced, i.e.: 0, 25, 50, 75, and 100. To satisfy the requirement of 5 unique values, at least 5 .libsrc files will be generated. If one of the other variables requires a larger number of unique values, then more .libsrc input files will be generated. Since none of the items in the above example script file requires more than 5 unique values, 5 .libsrc files will be generated by this script file.

The algorithm for creating a test ADRPM input file, in pseudocode is:

For each item in the script file,
Do any unused unique values still exist?

- if so, then randomly pick and write one of these unused values and mark it as used
- if not, then randomly pick and write one of the values

The algorithm for generating a group of files is:

Do any of the items still have unused unique values?
- if so, then generate another file
- if not, exit the program

These two simple algorithms used together generate a fairly small number of input files where each variable is exercised over its full range, and the variables are well "mixed" with each other. For an ADRPM input variable such as Wind, which can be None, Upwind, or Downwind, this method guarantees that all 3 values will be tested.

The next step is to perform a complete series of tests using all possible combinations of the input files generated. For example, our primary test series had 5 source files, 5 propagation files, and 5 detector files, requiring 5x5x5=125 runs. The nature of this testing makes it surprisingly easy to find out which input variable values or interrelationships cause problems. For example, perhaps in all 5 cases where source file number 2 is used in a run along with detector file number 3, the program complains about a bad surface roughness value and refuses to complete the run.

Any number of unique spectral data sets can be required for the source spectrum, ambient noise spectrum, and detector spectrum. We chose the 5 spectra shown in Figure 4. While these may not be entirely realistic for each item (source/ambient/detector), they did provide a good mix of input values and results.

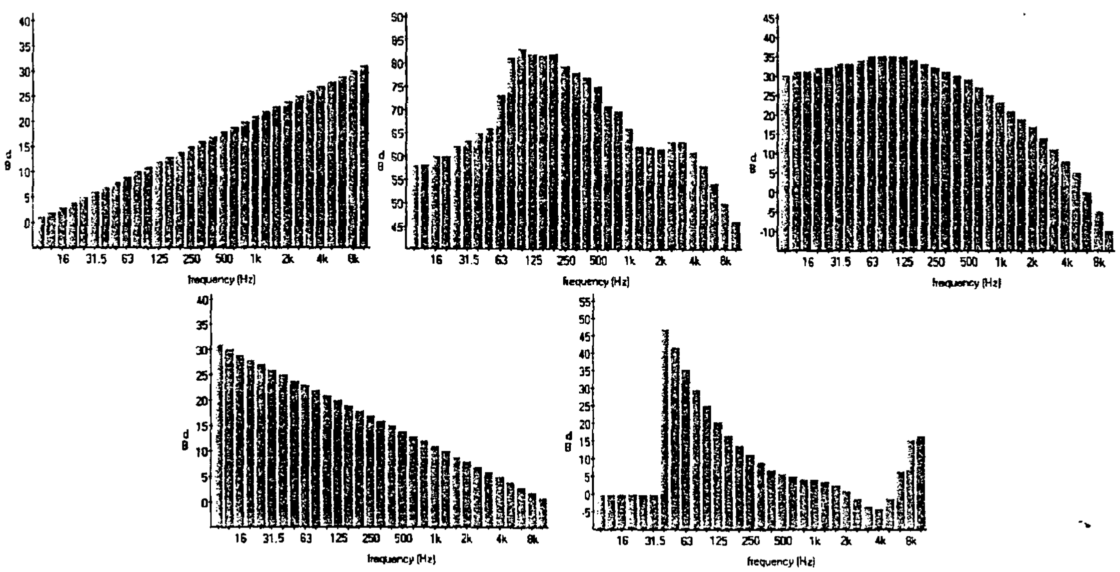


Figure 4 – Five Unique Input Spectra

Results of Testing

Table II shows the results of our first large test series, where all the runs were performed on both the PC/Windows version of ADRPM and the HP/Unix version. While 100% agreement between the two versions would have been a nice surprise, the results proved the value of this type of testing. While all previous human-operator testing had produced full agreement, this more automated testing brought out various problem areas for us to work on before pulling the plug on the old software. The testing also showed that most of the input value checking had been removed from the HP version, sometimes allowing invalid data sets to run. An example of an invalid data set is one where the RMS (root mean square) surface roughness exceeds the source height, microphone height, or detector height. The HP version lets many of these runs slip through, whereas the PC/Windows version refuses to execute them.

Possible Results of the Run	Number of Cases
Results match, or both programs indicate invalid inputs	60
Results don't match	4
PC version refuses to run due to invalid inputs, but HP version runs	35
Both versions crash due to illegal math operation	25
PC version crashes due to illegal math, HP version runs but provides questionable results	1
TOTAL	125

Table II – Summary of Test Results

The testing also provided result cases that we knew ADRPM could generate, but which we hadn't seen before or at best very rarely. These included multiple detection ranges due to barrier and/or foliage band effects, and cases where no solution could be found.

Future Developments

A desirable modification would be to allow a weighting function to be applied to the generated input values, instead of evenly spacing them over the legal range. We also would like to go further back in time and perform this analysis on the ancient (1985) DOS version of ADRPM. A complete rewrite of ADRPM starting with a "clean sheet of paper" remains a possibility, depending upon how well this new version is received by the ground vehicle signature modeling community.

Conclusions

This paper described the testing procedures used in comparing a new version of an acoustic detection model with the old one. The testing method exercises all of the input variables over their full ranges, but does not require an unreasonable number of runs. The use of scripts for specifying the input values and ranges provides flexibility. The testing proved very beneficial, as problems were discovered with both the new software and the old. The method described could probably be used for testing other similar modeling programs requiring large numbers of input values.

References

1: Fidell, S., Secrist, L., Harris, M., and Sneddon, M., "Development of Version 7 of an Acoustic Detection Range Prediction Model (ADRPM-7)", Technical Report 13397, Vols. I-III, U.S. Army Tank-Automotive Command, Warren, MI (1989)

2: Beizer, B., "Software Testing Techniques", Second Edition, Boston, MA: International Thomson Computer Press (1990)

3: Sommerville, I., "Software Engineering", Fifth Edition, Reading, MA: Addison-Wesley (1996)

Verification of a PC/Windows Version of the TACOM/TARDEC ADRPM Acoustic Model Translated from the HP-UX Version

Roger Evans, Robert Mantey, David Thomas

U.S. Army Tank-automotive and Armaments Command

Ground Target Modeling and Validation Conference
August 15-18, 2000
Houghton, MI

Introduction

- Software for modeling detectability of ground vehicle acoustic signatures
- Existing ADRPM model converted to PC/Windows platform
- New version of ADRPM needed to be tested vs. the old version

History of ADRPM

1970's - ADRPM I, BASIC

. . .

1980's - ADRPM V, FORTRAN

. . .

1990's - ADRPM VII, C

. . .

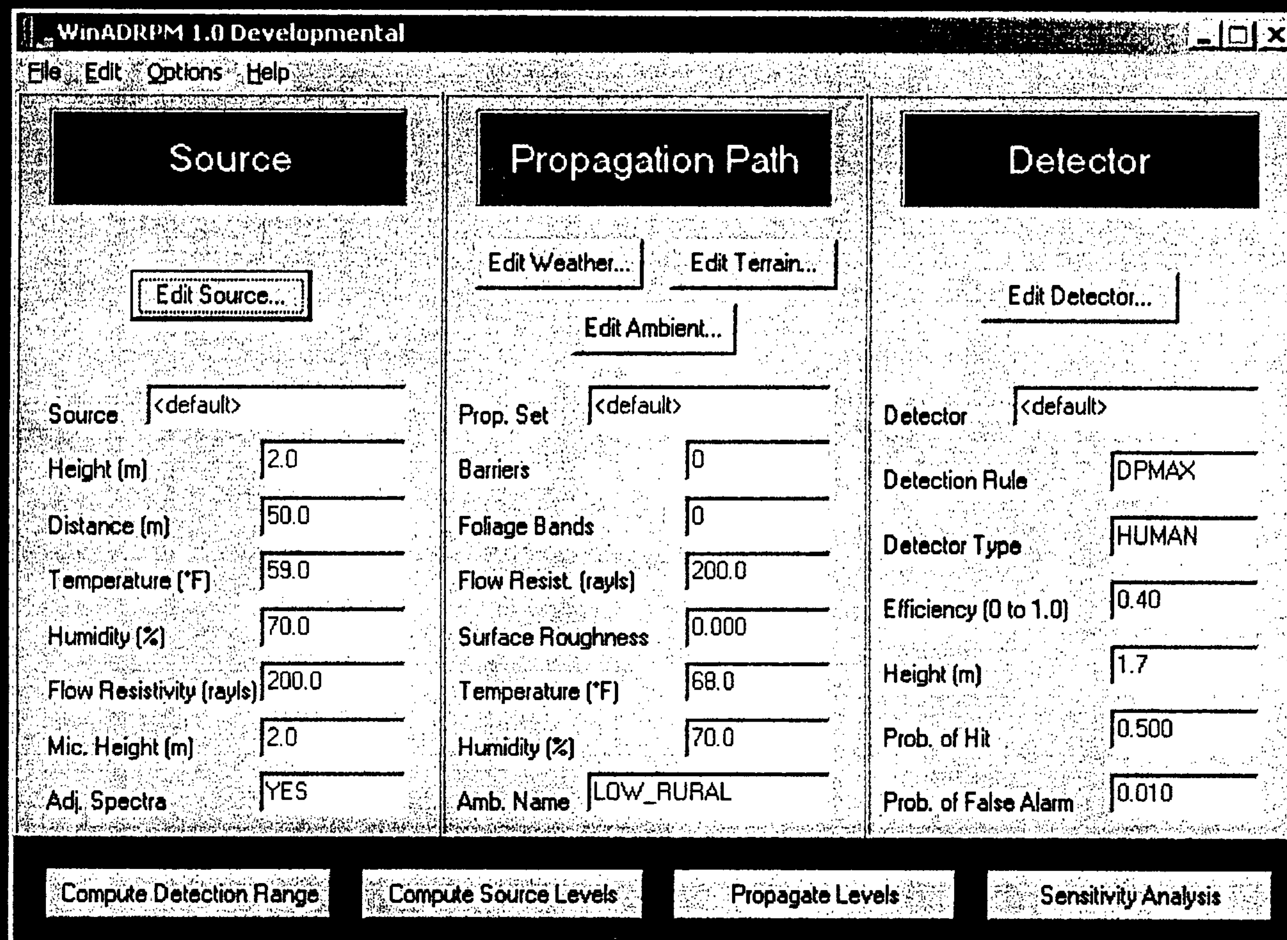
ADRPM Philosophy

- Keep it simple
- Usable by non-experts
- Ground-to-ground case
- Use algorithms only when they match real-world test data

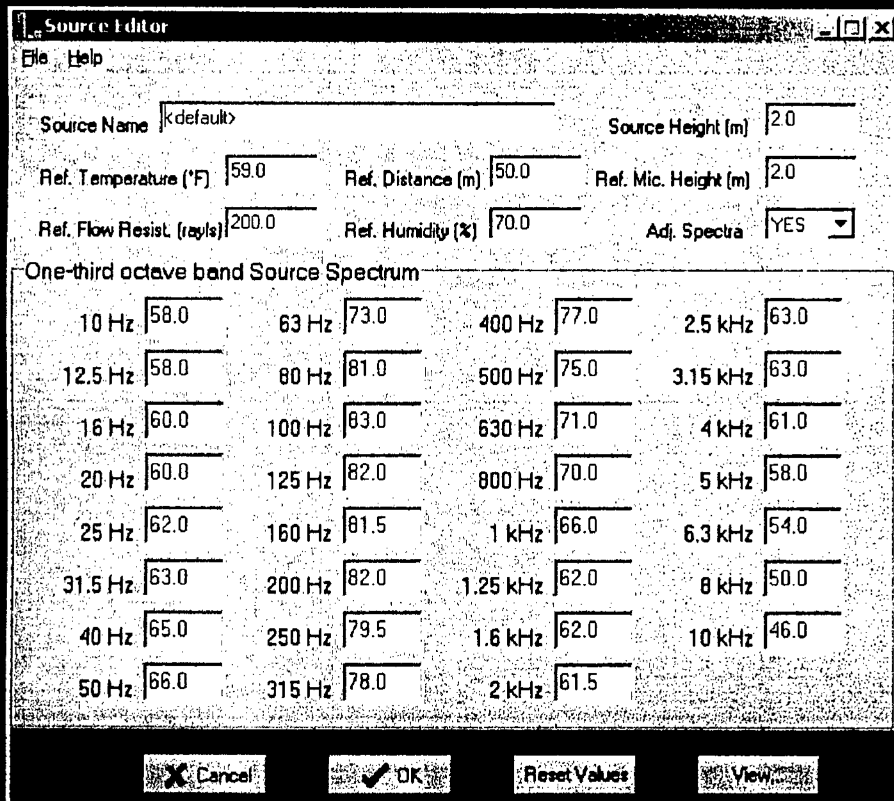
Important Comment

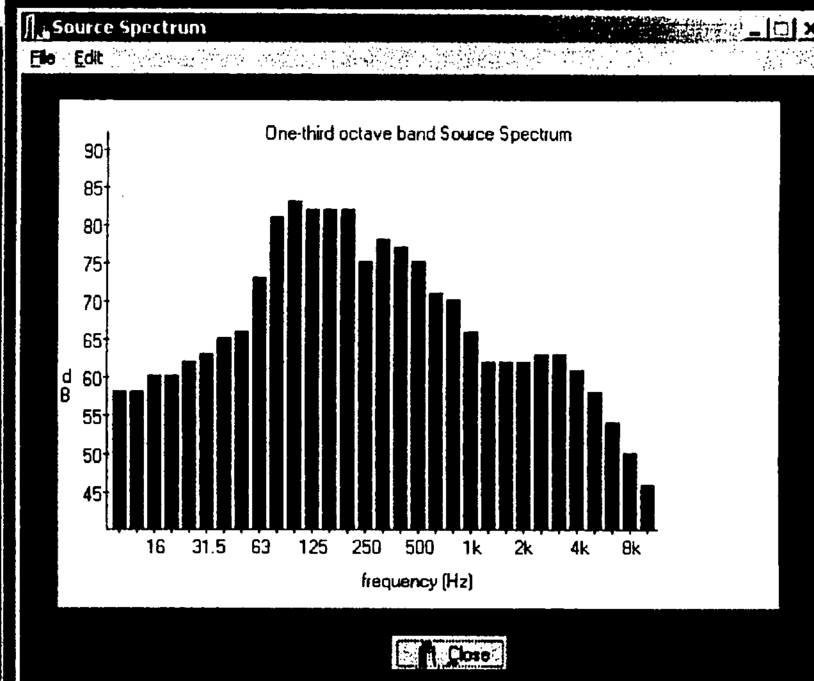
- ADRPM does NOT predict the acoustic signature of a vehicle
- ADRPM uses a measured or predicted signature to calculate detection ranges and detection frequencies

Source/Path/Detector are Clearly Separated in User Interface



Source/Path/Detector Have Similar Editors





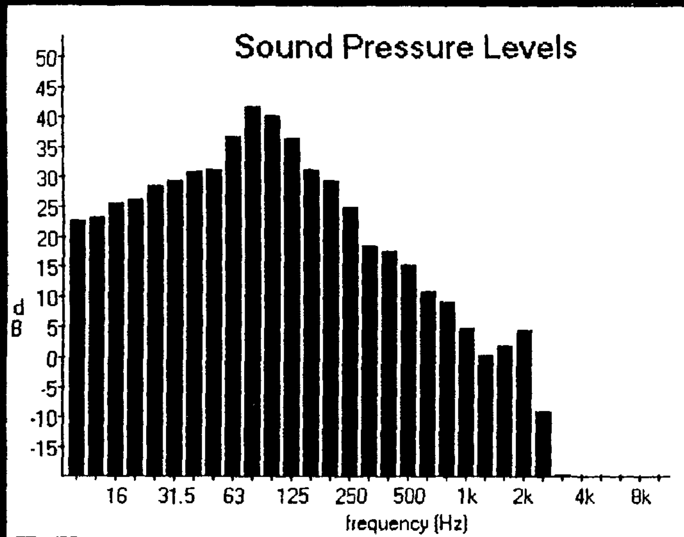
ADRPM Inputs

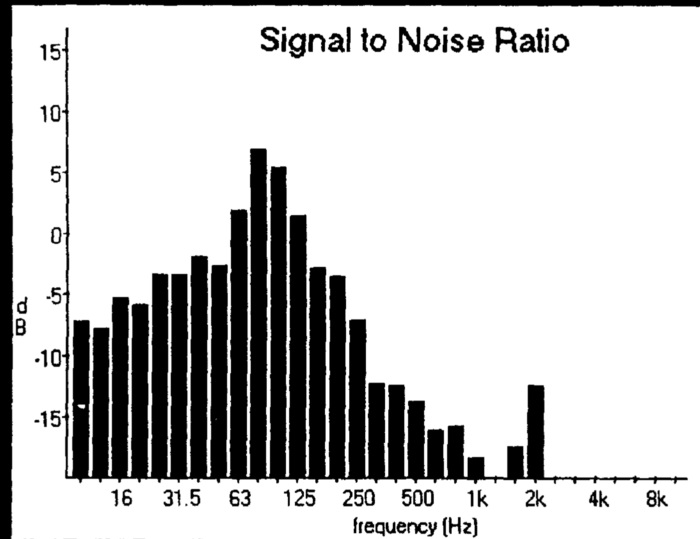
- Inputs include:
  - Source: spectrum, height, source-to-microphone parameters, environmental
  - Path: temperature, humidity, surface roughness, foliage bands, barriers, ambient spectrum
  - Detector: detection rule, type, efficiency, height, probability of hit, probability of false alarm, detector spectrum

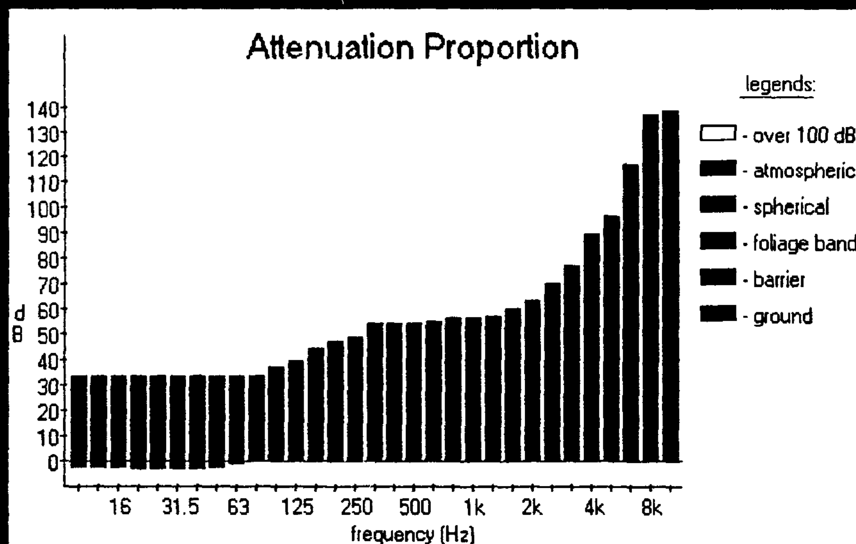
Four Primary Calculations

- Compute Detection Range: returns range and spectrum at that range
- Compute Source Levels: for a given range, returns most-detectable frequency, and maximum dB level for that frequency
- Propagate Levels: return spectrum at a given range, independent of detector
- Sensitivity Analysis: sweep a parameter over a specified range

Graphical Results for Each Calculation

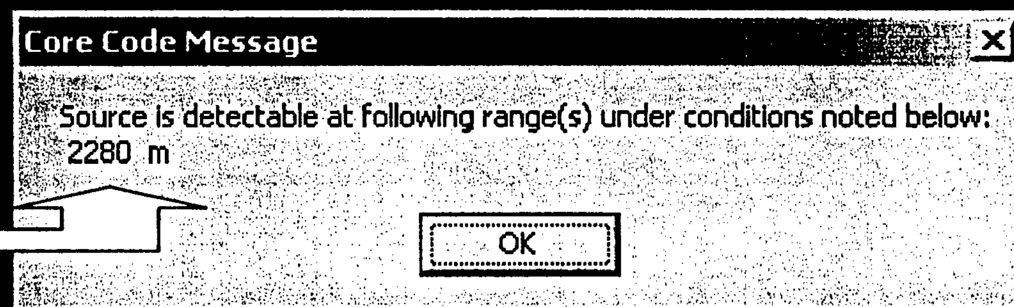


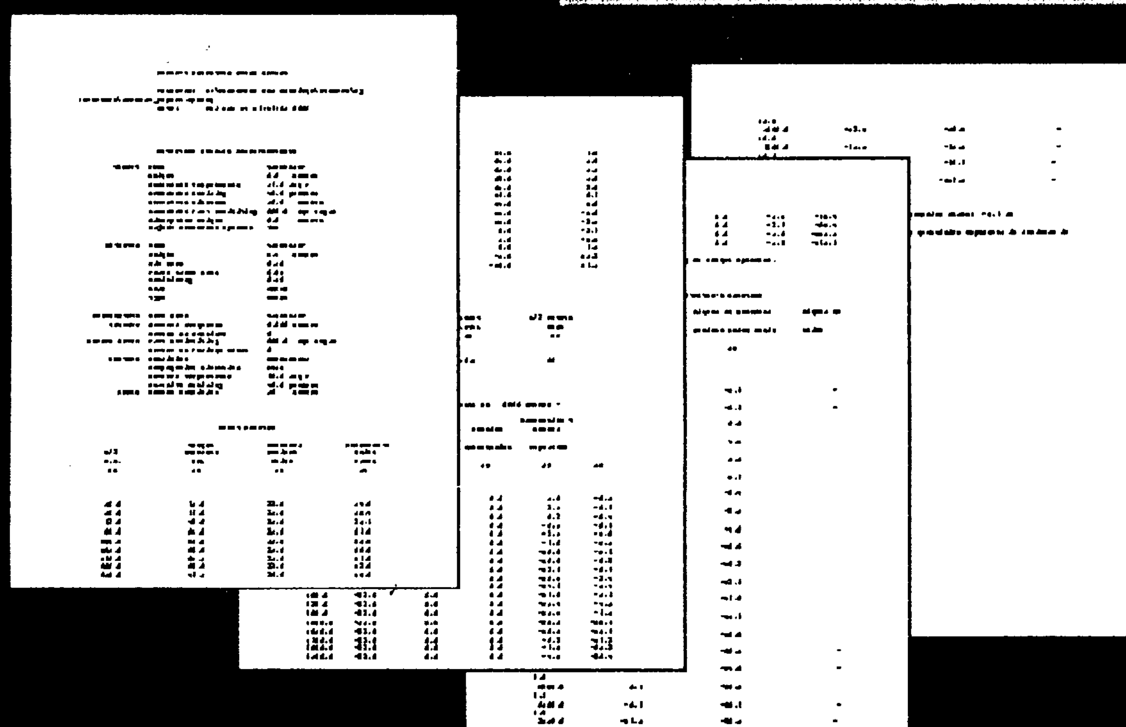




Also Textual Results

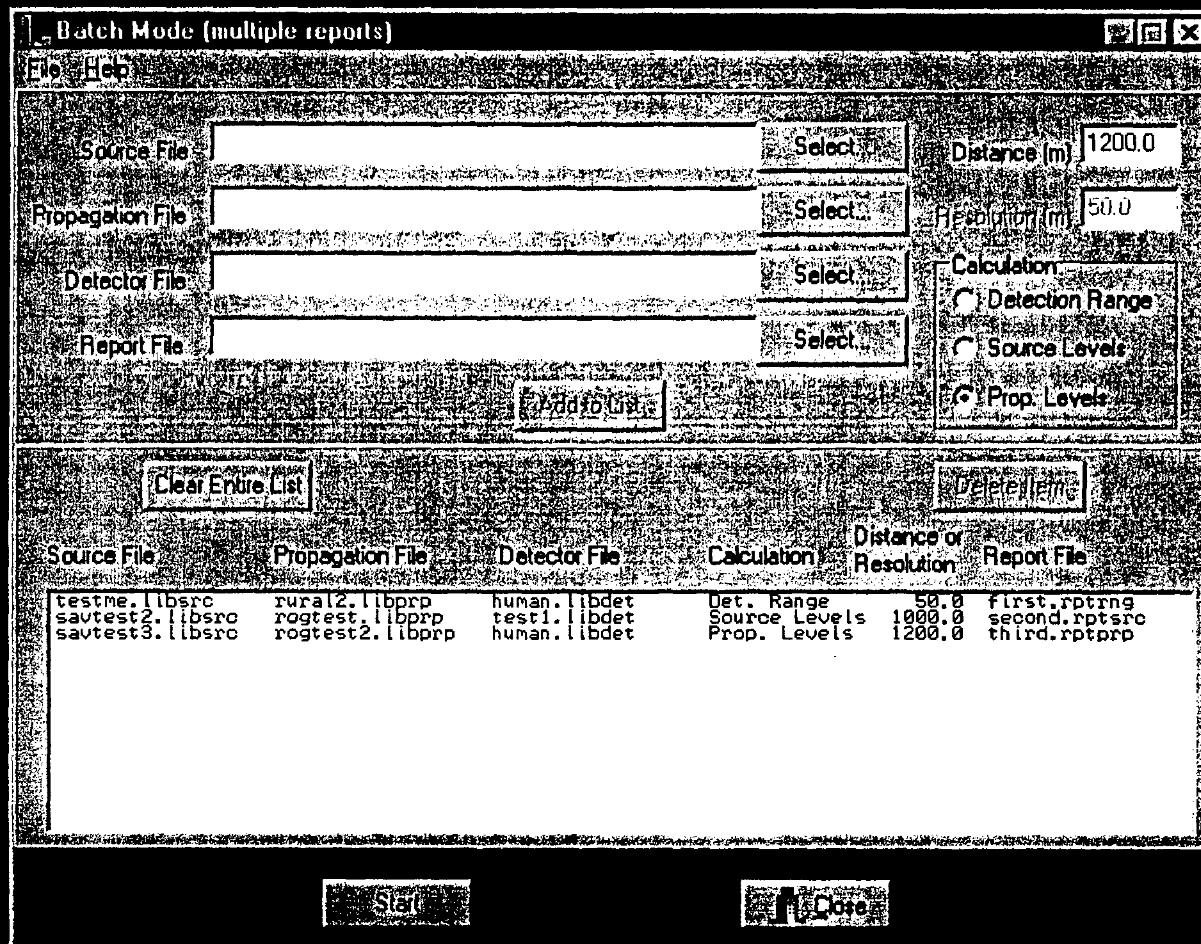
Detection Range





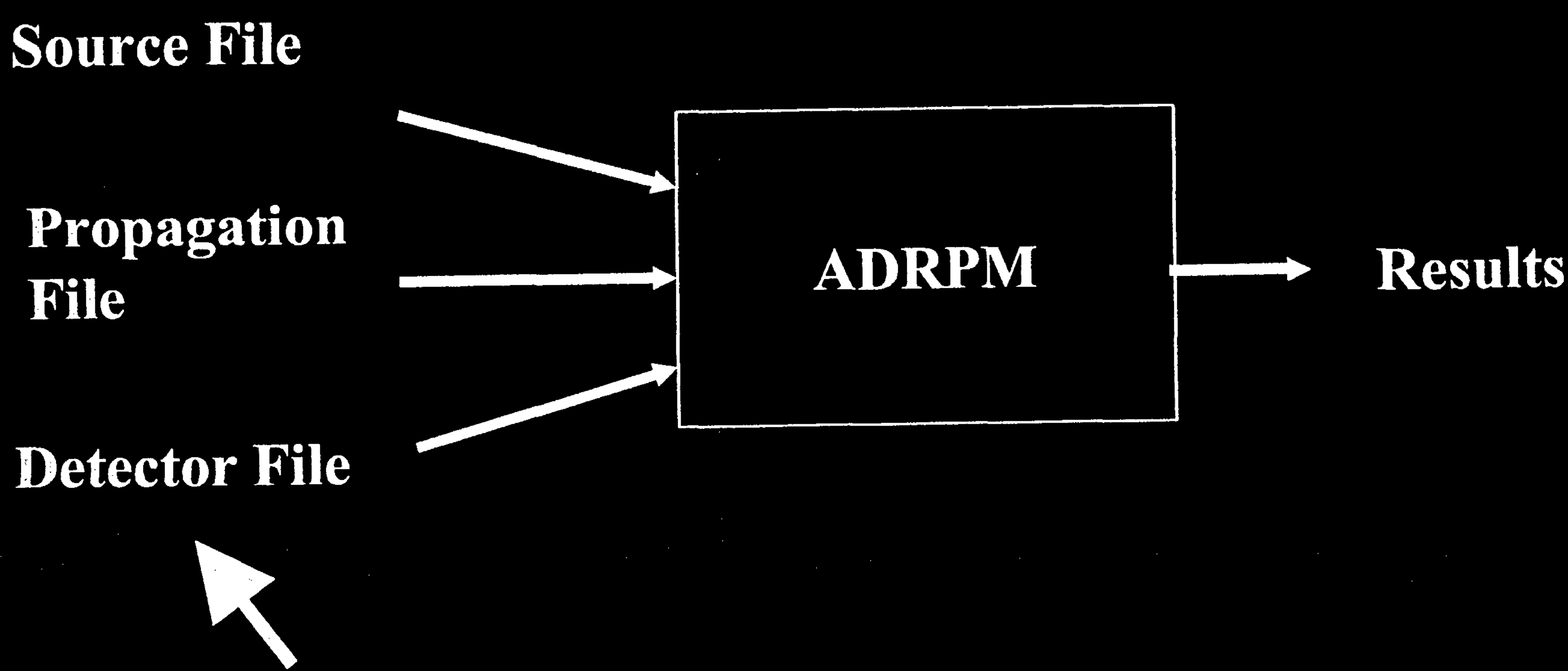
Reports

Batch Mode



Batch Mode (multiple reports)
File
Help
Source File
Propagation File
Detector File
Report File
Select...
Select...
Select...
Select...
Distance (m)
1200.0
Resolution (m)
50.0
Calculation
Detection Range
Source Levels
Prop. Levels
Add to List
Clear Entire List
Delete Item
Source File
Propagation File
Detector File
Calculation
Distance or Resolution
Report File
testme.libsrc rural2.libprp human.libdet Det. Range 50.0 first.rptrng
savtest2.libsrc rogtest.libprp test1.libdet Source Levels 1000.0 second.rptsrc
savtest3.libsrc rogtest2.libprp human.libdet Prop. Levels 1200.0 third.rptprp
Start
Close

Automated Generation of Test Cases



Test Case Inputs Being Generated

Method of Generating Test Cases

- Write a script file detailing range and number of unique values required by each input variable
- Program reads the script file and generates necessary number of ADRPM input files providing the number of unique values specified
- All values are randomly picked from lists, unused ones first
- Resulting ADRPM input files exercise all input variables over their specified range

Verification Runs

- Every possible combination of input files is tested
- Five of each input file type (source/propagation/detector) provides 5x5x5=125 runs
- Some runs will not execute due to invalid relationships among the variables -- this is OK!

ADRPM Test Results

Possible Results of the Run	Number of Cases
Results match, or both programs indicate invalid inputs	60
Results don't match	4
PC version refuses to run due to invalid inputs, but HP version runs	35
Both versions crash due to illegal math operation	25
PC version crashes due to illegal math, HP version runs but provides questionable results	1
TOTAL	125

Conclusions

- Testing "by hand" had not indicated any problems
- Automated statistical testing uncovered problems with both versions of ADRPM
- The generation algorithm developed for this testing exercises all variables over their full range, with a reasonable number of runs.

16254

OPSEC REVIEW CERTIFICATION

(AR 530-1, Operations Security)

I am aware that there is foreign intelligence interest in open source publications. I have sufficient technical expertise in the subject matter of this paper to make a determination that the net benefit of this public release outweighs any potential damage.

Reviewer: Wallace R. Mick Jr. GS-14 Mechanical Engineer
Name Grade Title

Wallace R. Mick Jr. 10 Aug 2000
Signature Date

Description of Information Reviewed:

Title: Verification of a PC/Windows Version of the TACOM/TARDEC Acoustic Detection Range Prediction Model Translated from the HP-UX Version

Author/Originator(s): ROGER EVANS

Publication/Presentation/Release Date: Aug. 16, 2000

Purpose of Release: Presentation on 16 Aug 00 at GTMV Conference and publication in the Unlimited Release Conference Proceedings

abstract, summary, or copy of the information reviewed is available for review.

Reviewer's Determination (check one) Very general modeling paper on acoustic detection & validation. No military data or sensitive content. Unclassified unlimited release is recommended. W. Mick

1. Unclassified Unlimited.

2. Unclassified Limited, Dissemination Restrictions IAW ________

3. Classified. Cannot be released, and requires classification and control at the level of ________

Security Office (AMSTA-CM-XS):

Concur/Nonconcur [illegible] [illegible]
Signature Date

Public Affairs Office (AMSTA-CM-PI):

Concur/Nonconcur [illegible] [illegible]
Signature Date

STA 7114-E
Sep 1999